Шаблон проектирования для реализации в практической работе №5: посетитель.

Обоснование выбора:

Как известно, нередко для разных категорий вкладчиков банки имеют свои правила: оформления вкладов, выдача кредитов, начисления процентов и т. д. Соответственно классы, описывающие данные объекты, тоже будут разными. Но что важно, как правило, правила обслуживания четко описает весь набор категорий клиентов. Например, есть физические лица, есть юридические, отдельные правила для индивидуальных или частных предпринимателей и т. д. Поэтому структура классов, представляющая клиентов, будет относительно фиксированной, то есть не склонной к изменениям.

Казалось бы, все просто, можно было бы и не использовать данный шаблон и каждый раз добавлять в Интерфейс IAccount новые методы и реализовывать их в классах Person и Company.

interface IAccount

{

    void ToHtml();

}

// физическое лицо

class Person : IAccount

{

    public string FIO { get; set; } //Фамилия Имя Отчество

    public string AccNumber { get; set; } // номер счета

    public void ToHtml()

    {

        string result = "<table><tr><td>Свойство<td><td>Значение</td></tr>";

        result += "<tr><td>FIO<td><td>" + FIO + "</td></tr>";

        result += "<tr><td>Number<td><td>" + Number + "</td></tr></table>";

        Console.WriteLine(result);

    }

}

// юридическое лицо

class Company : IAccount

{

    public string Name { get; set; } // название

    public string RegNumber { get; set; } // гос регистрационный номер

    public string Number { get; set; } // номер счета

    public void ToHtml()

    {

        string result = "<table><tr><td>Свойство<td><td>Значение</td></tr>";

        result += "<tr><td>Name<td><td>" + Name + "</td></tr>";

        result += "<tr><td>RegNumber<td><td>" + RegNumber + "</td></tr>";

        result += "<tr><td>Number<td><td>" + Number + "</td></tr></table>";

        Console.WriteLine(result);

    }

}

Но допустим после этого мы хотим внести еще много изменений в правила оформления и Person, и Company, но частое внесение изменений в фиксированную структуру классов не будет оптимально.

В роли абстрактного класса здесь выступает интерфейс IAccount. Однако его реализации теперь не содержат метод ToHtml(), и любой другой метод для сериализации в какой-либо формат. Так как вся функциональность по сериализации вынесена в отдельные классы посетителей. В итоге классы Person и Company становятся намного чище и проще по структуре.

И если нам надо добавить новый способ сериализации, достаточно просто определить еще один класс посетителя.